2013-04-19

Can a computer adaptive assessment system determine, better than traditional methods, whether students know mathematics skills?

Skyler Whorton
Worcester Polytechnic Institute

Can a computer adaptive assessment system determine, better than traditional methods, whether students know mathematics skills?

by

Skyler Whorton

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

April 25, 2013

APPROVED:

_____

Professor Neil Heffernan, Major Advisor

_____

Professor Joseph Beck, Thesis Reader

_____

Professor Craig Wills, Head of Department

Abstract

Schools use commercial systems specifically for mathematics benchmarking and longitudinal assessment. However these systems are expensive and their results often fail to indicate a clear path for teachers to differentiate instruction based on students' individual strengths and weaknesses in specific skills. ASSISTments is a web-based Intelligent Tutoring System used by educators to drive real-time, formative assessment in their classrooms. The software is used primarily by mathematics teachers to deliver homework, classwork and exams to their students. We have developed a computer adaptive test called PLACEments as an extension of ASSISTments to allow teachers to perform individual student assessment and by extension school-wide benchmarking. PLACEments uses a form of graph-based knowledge representation by which the exam results identify the specific mathematics skills that each student lacks. The system additionally provides differentiated practice determined by the students' performance on the adaptive test. In this project, we describe the design and implementation of PLACEments as a skill assessment method and evaluate it in comparison with a fixed-item benchmark.

Acknowledgements

Isaac Newton once said, “If I have seen further it is by standing on the shoulders of giants.” I owe the completion of this project to the gracious support of many such giants:

I must first thank Corey Belhumeur (MS candidate, WPI) for his sustained diligence and cooperation to develop the PLACEments feature in ASSISTments.

To evaluate PLACEments assessment and remediation, we worked closely with Kim Kelly (PhD candidate, WPI) and her students served as the first participants. I would additionally like to recognize Barbara Delaney, Janine Ohnemus and Jeffrey Grupposo (mathematics teachers in Massachusetts) and their students for their participation in additional evaluation trials.

For their early intellectual contribution to PLACEments and for their lasting inspiration I also want to thank Andrew Burnett (ASSISTments trainer) and Doran Smestad (WPI, 2014).

The sound design and smooth deployment of PLACEments would not have been possible without the guidance and discretion of David Magid, senior software developer of ASSISTments.

For advising this research and for creating the ASSISTments system, I wish to thank Professor Neil Heffernan and Cristina Heffernan.

This research was primarily funded by the National Science Foundation (NSF) GK-12 grant. Its implementation at WPI, Partnership In Mathematics and Science Education (PIMSE), is the fellowship through which I was able to conduct this research. Additional funding was provided by the Department of Education, the Office of Naval Research, and the Spencer Foundation.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

With the advent of the Common Core State Standard (CCSS) initiative and recent state legislation being passed in the US, it is of great interest to mathematics teachers to adopt new techniques to meet reporting and accountability requirements for each standard (i.e., skill) and for each student's growth. [15] Teachers therefore perform regular "benchmark" tests to track students' mastery of individual skills, but a traditional fixed-item benchmark test fails to diagnose the weaknesses underlying students' demonstrated lack of specific skills. This project was thusly motivated and a new system designed with the intent to address these needs. In this section, we discuss the research platform, ASSISTments, used for our proposed system and then describe the current state of the art in adaptive assessment.

1.1 ASSISTments

The ASSISTments platform is a web-based Intelligent Tutoring System (ITS) which derives its name from the words "assistance" and "assessment" due to the interactive nature of its use in the classroom. While students complete assigned work in ASSISTments, they may have immediate feedback on the correctness of their answers and may also exercise the option of requesting assistance in the form of scripted hints or worked examples known as scaffolding. These types of assistance are collectively referred to as "tutoring." Teachers are furthermore able to track students' progress in real-time and drive formative assessment in the classroom. [1]

One notable feature of ASSISTments is the Automatic Reassessment and Relearning System (ARRS). When used, ARRS tracks each student's performance as they complete specialized assignments called Skill Builders. Each Skill Builder randomly presents a small set of items drawn from a large pool of possibly hundreds, all based on a common template focusing on one specific mathematics standard ("skill"), e.g. *Dividing Whole Numbers*. A student can only complete the skill builder by correctly answering a certain number of consecutive items, e.g., three items in a row. After completing a Skill Builder, ARRS periodically assigns individualized reassessment items from the same content template as well as relearning items for skills that the student appears to have forgotten.

This feature has been shown to help maintain high levels of skill mastery [2] and anecdotally to increase teachers' awareness of specific gaps in student knowledge. However using Skill Builders as a means of benchmark assessment requires a relatively large time investment per skill. Attempting to benchmark a large number of skill proficiencies among a group of students using only Skill Builders would be time consuming for both the student and the teacher due to amount of effort and logistical overhead.

1.2 Student Mathematics Assessment

Assessing student knowledge is a large area of research interest. Of particular importance is the modern paradigm of Item Response Theory (IRT) [3] and its technological consequence, Computerized Adaptive Testing (CAT). [4] CAT systems based on IRT are often designed to produce a single "unidemensional" value to represent all of a student's knowledge of a subject. However, this monolithic scoring metric is assumed to be less useful to educators than differentiated diagnostic information, i.e., detailed metrics for the individual skills of each student.

To this point, we consider existing assessment methods for mathematics benchmarking in public schools, such tools as MAP (Measures of Academic Progress) [5] and Galileo K-12 [6], which are both examples of CATs geared towards K-12 education. These methods are discussed by Militello and Heffernan as being appropriate for district and state-wide benchmarking and longitudinal assessment. However, teachers reported their preference of using ASSISTments in the classroom rather than these alternatives because of the availability of detailed, item-level diagnostic information. [7] This anecdotally suggests that in order to address specific weaknesses, students and teachers may benefit more from individual skill data than traditional, summative metrics.

Related work has been done to use Bayesian student modeling, e.g., Knowledge Tracing (KT) in a CAT system to predict the probability of "skill" knowledge based on student scores across several homogeneous items. [8] However, the skills measured have historically been broad proficiency categories and/or require assessment over many items in order to estimate ability levels, rather than fine-grain skills. Moreover, the requirement of several items per skill in KT limits its utility in a short-duration, broad-stroke benchmark assessment covering as much of a curriculum as possible.

We believe that, rather than testing a student on every mathematics skill in their curriculum, a shorter and more detailed skill-level assessment may be possible by requiring at most one test item per fine-grain skill. Critics of this approach may claim that an assessment with this structure lacks the predictive power associated with a correlation between the CAT assessment results and high-profile summative assessments such as standardized state tests. However, our goal is not to create a predictor for summative assessments, but rather, to provide an assessment to show and to be informed by the individual skills possessed by examinees. To our knowledge, no CAT system exists that satisfies this objective in the manner proposed.

2 PLACEments

We propose an extension to the ASSISTments system called PLACEments that consists of:

- Technical considerations for processing a directed acyclic graph of interdependent skills and for those skills' associated items.
- An adaptive test algorithm implemented as a new "navigator" type in ASSISTments.
- An adaptive remediation algorithm similar to ARRS, but which is informed by both the students' performance on the adaptive test and by the skill graph.

2.1 Skill Dependency Graph

The concept of modeling student mathematics knowledge in terms of discrete skills connected by hierarchical, prerequisite relationships has been studied and implemented in various forms in contemporary work [11, 12, 14]. We assume here that Common Core State Standard skills may be encoded with their dependencies through the use of a directed acyclic graph. This graph is the main data structure used by the PLACEments test and remediation.

To model a set of mathematics curriculum graphically, each discrete skill is considered single, unique node. Since the two entities hold a one-to-one correspondence in this design, the terms "node" and "skill" are used here interchangeably. In our design, each node has two metadata properties: its name, e.g., "Dividing Fractions," and a numeric representation of the grade level ("year") and academic month ("month") in which it is taught according to the teacher's curriculum. For example, "7th grade in September" would be encoded as "7.1". Such representations are henceforth called "levels." Levels, i.e. determinations about when to teach a given skill, are not themselves standardized by the CCSS Initiative and are therefore subject to each teacher's or school's own judgment. The reason for this numerical encoding is a matter of implementation of the adaptive test procedure.

Dependency relationships are expressed by directed edges between skills. The pedagogical assumption is that if a student knows a given skill, they must have already known all of the immediate parent skill(s) or "prerequisites." Since this property is transitive, they are in fact assumed to have learned each of the skill's ancestor skills in the graph.[*] Conversely, if a student has not learned a given skill, they must not truly know any of that skill's child or "postrequisite" skills. Again, this property is transitive such that all subsequent postrequisites are assumed to be unknown to the student. Expressing these relationships using directed edges has the consequence that any node with out-degree zero has no postrequisites and any node with in-degree zero has no prerequisites.

[*] However this property results in potentially confounding test behavior in one case described in Section 2.2.

The edges between skills are additionally constrained such that no edge may lead from a node with higher level to a node of strictly lower level, i.e., a prerequisite skill's level must be less than or equal to each of the postrequisite skills' levels.

Finally, it may be noted that the entire skill graph used to represent a mathematics curriculum may itself be (and typically is) composed of a union of many unconnected sub-graphs. The implication is that some groups of skills are closely related, but may share no remarkably related pedagogical roots.

2.2 Adaptive Test Algorithm

A PLACEments test is initialized with two levels: an upper bound and a lower bound. The initialization algorithm uses these limits to respectively determine the initial skills (and the order of appearance) to be tested and the worst-case termination criteria, as detailed in the in Figure 1.

First, the graph of skills to be tested, G^*, may be a subset of the entire graph stored in the system, G. Therefore line 1 must narrow the scope of the test to only those skills with level between the *UpperBound* and *LowerBound* provided as user parameters to the test. Line 2 populates T, the "test queue," with all of the skills in the graph subset that lack post-requisite skills. This comprises the initial set of skills that every student is guaranteed to be tested on. Note that line 3 sorts these skills in descending order of level such that the latest skills will be tested first. The loop in lines 5 and 6 populates the "backtrack list," B, which is the set of skills that could potentially be tested if the student fails to demonstrate knowledge of any of the skills in T. Therefore B is populated by all of the pre-requisites of skills in T. (For our purposes , "pre-requisites" refers only to adjacent ancestors in the graph.) Line 8 returns a tuple, *Test*, consisting of G^*, T and B. Additional sets K and U begin empty and will be populated with the "known" and "unknown" skills as the student responds to items in the test system.

	InitializeTest(*G*, *UpperBound*, *LowerBound*)
1	$G^* \leftarrow$ "inner" partition of G cut by *UpperBound* and *LowerBound* according to level
2	$T \leftarrow$ subset of nodes in G^* having out-degree 0 within G^*
3	sort T in descending order of level
4	$B \leftarrow \emptyset$
5	for each skill $t \in T$:
6	add to B all prerequisites of $t \in G^*$, removing duplicates
7	$K, U \leftarrow \emptyset$
8	return $Test \leftarrow \langle G^*, T, B, K, U \rangle$

Figure 1 – Adaptive test initialization algorithm

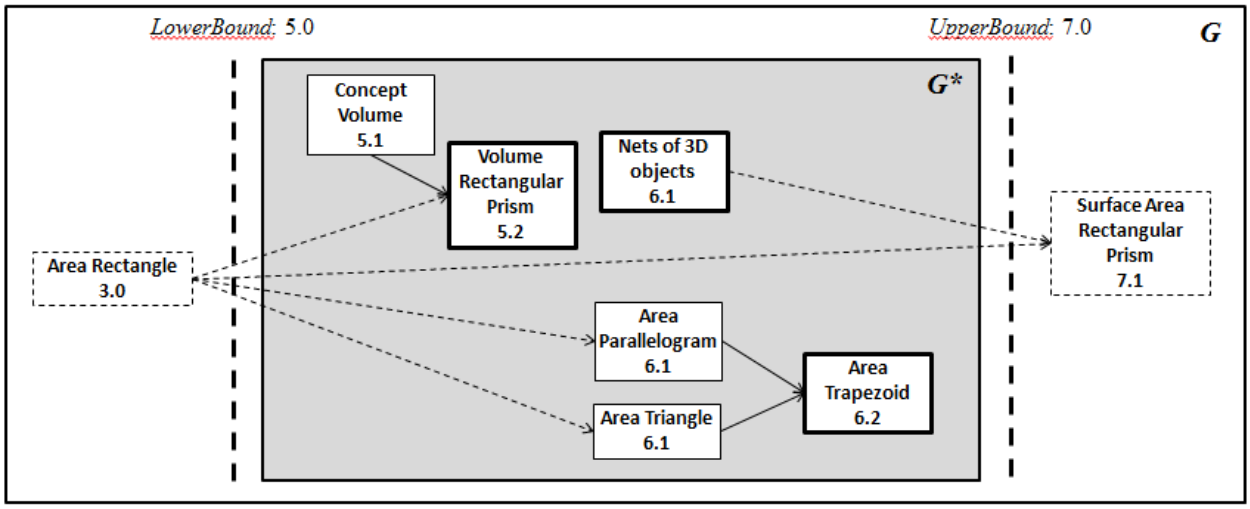


Figure 2 - Skill subgraph used in initialized adaptive test

An example of an initialized test's skill dependency graph is shown in Figure 2. The entire graph *G* contains two skills that were partitioned out of the test subset by cuts at levels 5.0 and 7.0 (determined arbitrarily here for the purpose of example). In practice, the *UpperBound* ought to reflect the teacher's decision about the level which the test taker should consider their goal. The *LowerBound* indicates the lowest skills which the teacher would like students to be tested on if they fail on post-requisite skills.

These nodes and their adjacent edges are dashed to indicate that they will not be observed by the test algorithm. The test subset, *G**, is shaded gray to show that the nodes inside may be used by the test. Nodes with effective out-degree of zero have a bolded border to show that they will begin in *T*. Their pre-requisites are the remaining nodes and these skills will begin in *B*. The resulting test queue and backtrack list are shown in Table 1.

Test queue, *T*	**Backtrack list, *B***
Area Trapezoid (6.2)	Area Parallelogram (6.1)
Nets of 3D Objects (6.1)	Area Triangle (6.1)
Volume Rectangular Prism (5.2)	Concept Volume (5.1)

Table 1 - Initial test queue and backtrack list of a sample test

Once a test has been initialized, it may be used to assess a student's skills. The system implementing the test is presumed to have a mechanism for administering an item corresponding to the selected skill, *S*, and for reporting the binary correctness, *C*, of the student's response to that item. (The way we achieve this in ASSISTments is given in 2.4) The process for selecting a skill from the adaptive test bank is simple: pop the next skill from the head of the test queue. Figure 3 details the algorithm that reacts to the student response received by the implementing system.

The first step in updating the test is to remove the tested skills *S* from the testing queue, in line 1. The conditional block from lines 2-7 then adds the skill to either the known or unknown sets depending on whether the student's response to the corresponding item was correct.

If their response was correct, the skill is added to the known list, *K*, in line 3. Any previously untested ancestors of that skill are also added to *K* in line 4. It is important to note that this behavior can result in confounding cases if some ancestor is added to *K*, but then later one of its post-requisites is tested and the item is responded to incorrectly. Should all of the pre-requisite skills of that item be tested, even the one(s) we inferred to already be known due to performance observed on other items? Normally the pre-requisite skills for the incorrect item would be tested, but in this case, the previously inferred "known" ancestor will not be tested. This was a design decision made in the interest of simplicity but could be reexamined in future work.

If their response was incorrect, line 7 will add to the test queue all of that skill's prerequisites that haven't already been tested. Lines 8-9 rebuild the backtrack list, *B*, from the empty set and include only the prerequisites of skills in the updated test queue, *T*, which haven't already been tested, i.e., exist in the known or unknown sets, *K* and *U*.

	UpdateTest(*Test*, *s*, *c*)
1	remove *s* from *Test.T*
2	if *c*:
3	add *s* to *Test.K*
4	add to *Test.K* each ancestor, *a*, of $s \mid a \in G^*, a \notin \{ Test.K \cup Test.U \}$
5	else:
6	add *s* to *Test.U*
7	push all pre-requisites of $s \in Test.B$ into *Test.T*
8	$Test.B \leftarrow \emptyset$
9	add to *Test.B* all pre-requisites of $Test.T \notin \{Test.K \cup Test.U \}$
10	return *Test*

Figure 3 - Adaptive test updating algorithm

After each time the test is updated, the implementing system must check whether the test is finished before continuing. In this design, the test is terminated if and only if *T* is empty immediately after the UpdateTest algorithm is completed. It follows that *B* will also be empty under these conditions, but *B* could be empty even if *T* is not in the case that all members of *T* do not have any prerequisites in *G**.

The sample test from Figure 2 is depicted again in Figure 4 after a student has answered the first item correctly. They have therefore demonstrated explicit knowledge of "Area Trapezoid" and implicit knowledge of its prerequisites. The updated *Test* tuple is shown in Table 2.

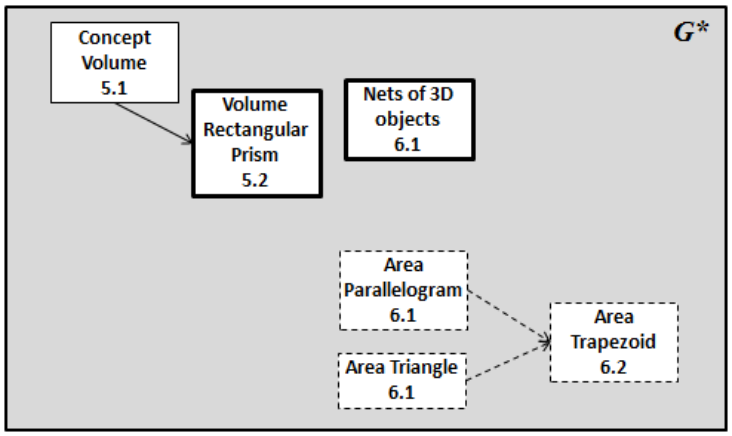


Figure 4 - Skill subgraph after updating for a correct student response

Test queue, *T*	**Backtrack list, *B***	**Known skills, *K***
Nets of 3D Objects (6.1) Volume Rectangular Prism (5.2)	Concept Volume (5.1)	Area Trapezoid (6.2) Area Parallelogram (6.1) Area Triangle (6.1)

Table 2 - Test queue, backtrack list and known skills list after updating sample test

2.3 Adaptive Remediation Algorithm

Research to evaluate the effectiveness of the PLACEments adaptive remediation as a learning aid is proposed in [9] and is currently being conducted. We focus here only on its general design and behavior in order to inform the portion of our assessment analysis concerning remediation completion data.

The process of a student completing a test as described in Section 2.2 results in a tuple of $\langle G^*, \emptyset, \emptyset, K, U \rangle$ where the empty sets are simply the depleted test queue and backtrack list. The goal of adaptive remediation is to provide each student with practice work corresponding to the set of unknown skills, *U*, detected by the adaptive test through incorrect student responses. This is a reoccurring process that starts immediately with the procedure given in Table 3 after the student completes a PLACEments adaptive test.

This initialization algorithm further narrows G^* down to only the subset of the graph of skills that the student did not know. The procedure then populates a new set A of "assigned" skills for remediation starting with the most basic, i.e., no more basic pre-requisites exist within the confined G^*. We assume here that the remediation system will create, provide and/or assign the practice work corresponding to the skills in A as necessary following initialization.

	InitializeRemediation(G^*, K, U)
1	$G^* \leftarrow$ subset of G^* that includes only nodes in U and edges between such nodes
2	$A \leftarrow \emptyset$
3	for each skill $u \in U$:
4	if u has in-degree 0 within G^*:
5	remove u from U
6	add u to A
7	return $\langle G^*, A, K, U \rangle$

Table 3 - Algorithm for initializing remediation work following a PLACEments test

Whenever a student checks the remediation system, e.g., after completing some remediation work, we expect that the system will automatically invoke the UpdateRemediation procedure given in Table 4. This procedure inspects all skills in A and updates the list of known skills, K, if new remediation work was completed. This also results in removing that skill from A. The update procedure then inspects each post-requisite of the newly "learned" skill and if all the pre-requisites of a post-requisite are already known, then that skill is moved from U to A. In other words, yet-unknown post-requisites are only assigned if all of their pre-requisites are known. We assume that the remediation system will create and assign new remediation work each time the elements of A are changed in this manner.

	UpdateRemediation(G^*, A, K, U)
1	for each assigned skill $a \in A$:
2	if remediation work for a is complete:
3	remove a from A
4	add a to K
5	for each post-requisite p of a :
6	if $p \in U$ and $p \in G^*$:
7	if $Pre \subseteq K$ | Pre is the set of pre-requisites of p:
8	remove p from U
9	add p to A
10	return $\langle G^*, A, K, U \rangle$

Table 4 - Algorithm for updating current remediation work

2.4 Implementation in ASSISTments

ASSISTments is a mature platform with over eight years of concerted development history and dozens of incrementally added features sharing the same infrastructure. It consists of a client-side web application and a server-side database interface and processing libraries. Both client and server are regularly maintained and updated, therefore it was a straightforward matter to design and implement the PLACEments adaptive test and remediation.

2.4.1 Storing the Skill Graph Data Structure

The skill graph data type described in Section 2.1 was realized in the ASSISTments database in preexisting relations. Skills themselves are tuples in the *skills* relation in the form ⟨*name*, *level*, *transfer_model_id*⟩. The "transfer model" is a historical term in ASSISTments for the manner by which skills are grouped into different versions of independently created skill dependency graphs. For example, [12] discusses multiple transfer models representing mathematics skills at different levels of granularity. Dependencies (edges) between skills are stored in the relation *prerequisite_skills*. Each tuple in this table is a simple pair of ⟨*parent_id*, *child_id*⟩ such that these attributes represent the primary key of the pre- and post-requisite skills, respectively. Items in ASSISTments are tagged with their corresponding skills in the relation *problem_to_skill_associations* which we simplify here to the form ⟨*problem_id*, *skill_id*⟩. The cohesive structure of "problems" in ASSISTments is irrelevant to the implementation of PLACEments except to note that the skill metadata for a problem is always available on both the client and server applications.

2.4.2 The PLACEments Navigator

In ASSISTments, the software component running on the client system is known as the "tutor." The tutor is a modern, asynchronous web-based application developed in Java and compiled into Javascript via the Google Web Toolkit (GWT). Among a fairly complex set of infrastructure components in the tutor is an abstract class called *Navigator* which is implemented for each type of test procedure desired in the tutor. For example, the most commonly used test procedures are *Linear* and *Random*, whereby items from the test bank are either presented in a static or dynamically randomized order, respectively.

We created a new *PlacementsNavigator* which implements the adaptive test algorithms described in Section 2.2, complete with its input parameters of a skill graph and upper and bound lower bounds on included levels. The *Test* tuple from these algorithms is more or less stored and managed by a separate *SkillQueue* object, which is totally informed of the state and related properties of the test while remaining ignorant of the tutor system internals.

Since the tutor system is "stateless" and based on asynchronous events, only persisting in the database (rather than on the client side) information about student responses, it was necessary to account for the possibility that a student may start and finish a PLACEments test in two or more separate sittings. The *SkillQueue* class contains a method by which the students' sequence of actions stored on the server may be replayed through the adaptive algorithm as if they were occurring in real-time. This effectively reconstructs the student's progress as it was when they last worked on the test. However it is still our intention for a PLACEments test to be administered and completed in only one sitting whenever possible.

Like all student responses to items delivered by the tutor application, each student's activity in their adaptive test items is sent from client to server and persisted as a single tuple in a relation called *problem_logs*. When a student starts (and completes) the adaptive test, they additionally persist an update of their own tuple (representing their progress on the test) in the relation *reaction_assignments*, which handles many types of logs including PLACEments adaptive tests and remediation work.

2.4.3 Assigning Remediation

As soon as a student finishes a PLACEments adaptive test, the server application processes the responses to the items that appeared on their exam according to the InitializeRemediation algorithm discussed in Section 2.3. ASSISTments then consults a new database relation, *skill_sequences*, which is basically a lookup table of which practice work to provide the student for a given skill. In the current design, the remediation work is in fact comprised of Skill Builders as described in Section 1.1.

The "PLACEments remediations" appear in a separate section in the student's web application as shown in Figure 5. The student can open and work on remediations as if they were a normal assignment in ASSISTments. Each remediation assignment, when finished, triggers the UpdateRemediation procedure described in Section 2.3. This results in an updated interface similar to Figure 5 that offers only the next "unlocked" work corresponding to only the skills whose pre-requisites are all considered by the system to be known to the student. However, students are always aware of how many more (i.e., "8 remaining") remediations will be assigned in addition to those currently offered.

When all PLACEments remediations are completed, that section of a student's interface disappears, but nothing else remarkable appears to the student. Persistent in the database, however, are all of the student's logs for the completed work. In future refinements of the PLACEments system, we would like to use past performance on PLACEments tests and remediations to tune the initialization and scope of subsequence PLACEments tests.

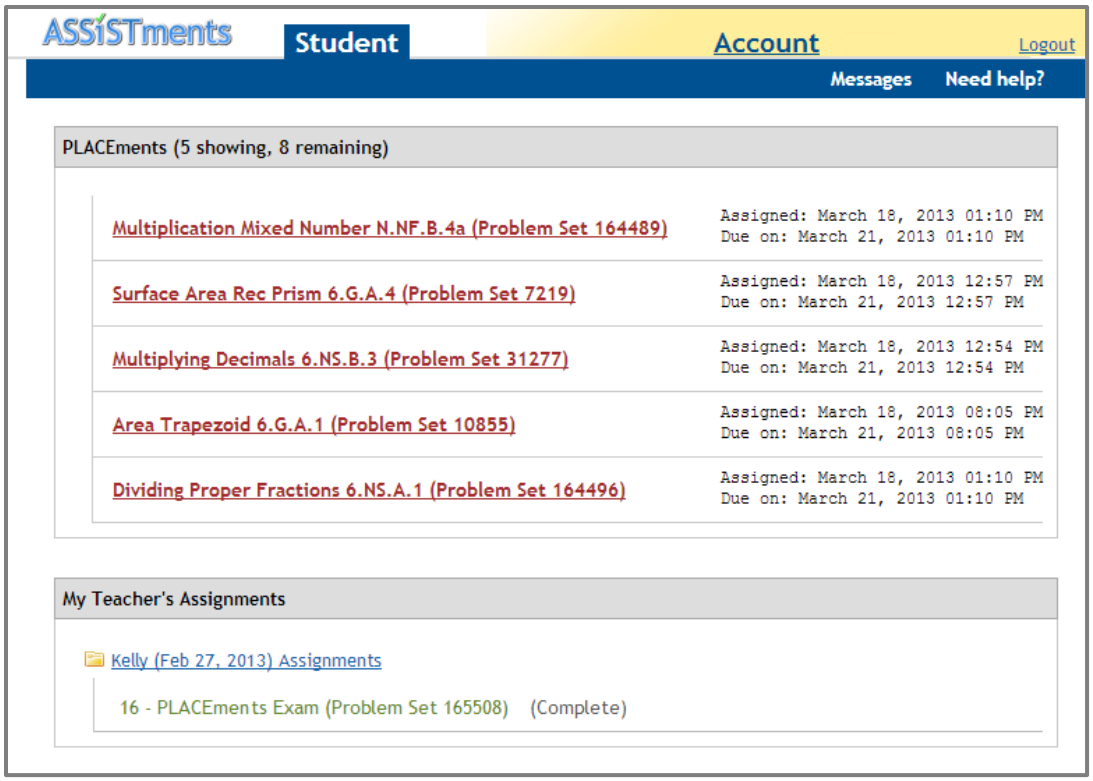


Figure 5 - Student's view of ASSISTments after initiating adaptive remediation

3 Assessment Evaluation

3.1 Overview

To evaluate the skill assessment accuracy of the PLACEments test, we designed a process in which a population of middle school students of various levels uses PLACEments. We compare their performance with results from a benchmark post-test after an interval of optional remediation. Section 3.2 provides the complete details of this process as well as assumptions made, biases and confounding factors. Sections 3.3 and 3.4 contain the experimental results and analyses including methods attempted to model skill knowledge based on PLACEments data.

3.2 Experiment Design

3.2.1 Guiding Questions

The central question to this experiment is, "how well does PLACEments predict specific skill knowledge in comparison with a fixed-item benchmark test?" However our adaptive system makes a fair

number of assumptions that leave us some room to ask additional questions about the effectiveness of our design, for instance:

- **How useful is reasoning on the skill dependency graph topology to a predictive model?** We must compare several variations of the collected data to analyze the impact of considering student knowledge of the pre-requisite skills of the benchmark test items, or not.
- **How may data from the remediation work be used to improve our predictive model?** We must include data about the completion of remediation work and compare such models to more basic models.

Another obvious question to be asked is, "what *effect* does completion of remediation work have on the students' post-test score?" However this line of research regarding gain is instead proposed and investigated in a parallel study in [9].

3.2.2 Participant Selection

A total of 173 students across two suburban Massachusetts schools, three grades and four mathematics teachers participated in the assessment evaluation study. The characteristics of the samples of students are detailed in Table 5.

We coordinated with the participating teachers to ensure that each student had as close to two weeks as possible between the pre- and post-test. The count of students here represents only those who completed both tests although other students not counted here may have taken one or the other due to absence, lack of diligence or poor coordination with their teacher.

Group	School	Grade(s)	PLACEments test date	Mean period between pre- and post-test	Students
I	A	6^{th}-8^{th}	3/18/2013	14 days, 11 hours	19
II	B	6^{th}	3/26/2013	13 days, 10 hours	78
III	B	7^{th}	3/27/2013	14 days, 1 hour	39
IV	B	7^{th}	3/27/2013	13 days, 22 hours	37

Table 5 - Participating student population in PLACEments assessment evaluation study

3.2.3 Experimental Skill Dependency Graph

The long term vision of PLACEments is for it to operate over a teacher's entire mathematics curriculum encompassing all skills. However as the first iteration of PLACEments used by real students, we chose to limit the scope of the tested curriculum to a subset of 24 skills comprising a substantial breadth of sixth and seventh-grade mathematics as well as some earlier skills.

We developed a custom skill graph based on the requested needs of the teacher of Group I described in Table 5, whose students are behind in their mathematics class and attend this teacher's class for remediation. The teacher and ASSISTments staff worked together to design a skill graph for which we believed the corresponding adaptive test would greatly challenge students between grades six and eight, but eventually end at very basic level of knowledge. The upper and lower level bounds were set to 7.2 and 3.0, corresponding to the maximum and minimum level among skills in the experimental graph. The full graph of 24 skills across four disjoint (i.e., unconnected) components is shown in Appendix A.

It is important to note that neither the graph's content nor its topology have been empirically vetted or used in other research, but rather, was inspired by several content experts' intuition. We therefore keep in mind some uncertainty about the accuracy and expressiveness of the dependency graph moving forward.

3.2.4 General Procedure

The teachers participating in the assessment evaluation studies were briefed ahead of time about the intention and use of the PLACEments adaptive test and remediation. Teachers were instructed to administer the test to their students in one sitting and urge them to complete it in a single period. The general procedure for running the evaluation study with each group is as follows:

1. Administer the PLACEments adaptive test to each participating student. Every student is tested on at least the seven core skills. In our analysis, this invariant takes the place of a control condition in which the students would each take a fixed-item benchmark test to assess the students' knowledge of these specific seven skills. However the students may be tested up to a maximum of 24 skills should they respond with mostly incorrect answers. This behavior is expected as a direct result of the adaptive test algorithm.
2. Allow a period of two weeks during which teachers instruct their students to complete as many PLACEments remediation skill builders as possible. The attitude and means by which each teacher approached this step was not directly observed and is therefore identified as a possible source of variation among the four groups in the amount of effort spent on remediation work.

3. Administer a uniform, fixed-item post-test in ASSISTments consisting of one item for each of the 24 skills in the experimental dependency graph. All students answered exactly the same items and in the same order.

3.2.5 Preparation in ASSISTments

Since PLACEments is still being integrated into ASSISTments at the time of this research, it was necessary in this study to perform several administrative tasks manually in the system to prepare for students' participation. This did not hinder our ability to give the test or to collect data, but we mention it here only to clarify that participating teachers were not in any way involved with the administrative aspects of the adaptive test.

The steps we took to prepare the tests were as follows:

1. Choose two items, estimating the appropriate level of difficulty, to test the corresponding skills. One of these was used in all instances of the PLACEments adaptive test. The other was used in the static, fixed-item post-test. The items were chosen by content experts, but not specifically examined for the empirical quality in the hands of student users.
2. Similarly, existing Skill Builders were chosen by content experts to serve as the remediation work for each of the 24 skills. As described in Section 1.1, a Skill Builder may randomly present different students with different items. Therefore there may be some variation in the level of difficulty among items within a Skill Builder attempted by different students. However since most items within Skill Builders are nearly identical in their form (containing only different values for variables rather than drastically different formats), we assume the differences among items are negligible.
3. Before students in each group could access the adaptive test or the post-test, we took responsibility for creating and configuring them in the system. Students were only able to access the tests starting at 8:00 AM on the schedule test day and were not limited in when they could complete it.

3.3 Results

3.3.1 Participation

The 173 students observed exhibited a wide range of ability levels. This grants us some assurance that our analysis is not limited to specifically high-achieving or under-achieving students, however the vast majority of the students were unable to answer all core items correctly. Their overall scores on the adaptive test are charted in Figure 6, however each student score may be out of a different number of

items. This explains the skew towards lower scores since a student is more likely to answer half or more of the items incorrect if they don't know the core skills in the dependency graph.

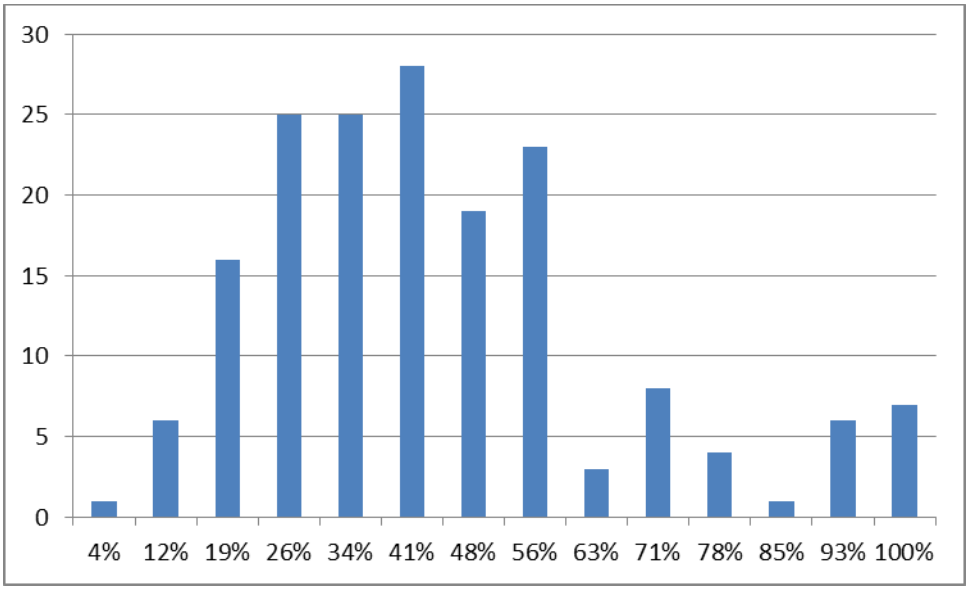


Figure 6 - Histogram of adaptive test score for entire student sample

3.3.2 Data Models

To analyze the assessment value of PLACEments, we created several data models to express various features of the students' performance. We separate the models created into two approaches: predicting student knowledge of the *core* skills in the graph, i.e., those desired benchmark skills with out-degree zero, and predicting the same for each skill in the graph.

Predicting Core Skills

We experimented with four slightly different datasets to weigh the predictive power of various features. Each instance in these datasets represents a single student's knowledge of one of the seven core skills.

1. Use 7 core pre-test items to predict 7 core post-test items. Uses only skill info and pre-test correctness.
2. Same as 1, but add student-identifying parameter.
3. Same as 1, but add total percent of the skill's prerequisites already known based on pre-test data.
4. Same as 3, but add total percent of remediation work completed for skill and its pre-requisites.

Predicting Each Skill

We also ask whether student performance on the PLACEments test can help predict their knowledge of all of the skills in the dependency graph, even if they weren't ever tested on them in the adaptive test. Corollary to this question is, "how reliable is the adaptive test's assumption that, if you test correct on an item, you must know its pre-requisite skills?" To explore this question, we created three additional datasets composed of various features but where each instance relates a student to their knowledge of one of the 24 skills.

5. Predict each of the 24 skills using skill data and pre-test correctness. Correctness of items for core skills is reported accurately. We manipulate the correctness for non-core skills to all be "correct," to simulate an assumption that may be made when administering a fix-item benchmark (that the students know the pre-requisites).
6. Same as [5], except that correctness of non-core skills is reported truly. This correctness is a ternary attribute consisting of "inferred," "incorrect" or "correct." Inferred in this case means the skill was not tested because its post-requisites tested positively.
7. Same as [6], but add remediation work data including: number of items completed in remediation, sum of time spent on remediation and ternary attribute telling whether "none was assigned," it was "completed" or is still "incomplete."

3.4 Analysis

We used Weka 3.7.7 to create and compare predictive models based on the seven datasets described in the previous section. To offer a baseline statistic of predicting student knowledge for a given skill, we used ZeroR (majority class) to show that the simplest models produce predictions scarcely better than a random guess. The OneR algorithm (split on one attribute) performs significantly better, but never reaches 70% classification accuracy.

In this analysis, it is appropriate to explore the use of both decision trees and a logistic regression. Decision trees are often advantageous when attempting to model disjunctive expressions and/or when the target class is discrete-valued, both of which are the case here. [16] A logistic regression is particularly appropriate for a binary-valued target class, as is the case in predicting binary knowledge of a skill or correctness of a test item. However our results showed that decision trees were marginally less effective than our logistic regression models in almost all cases, so we present only the logistic regression results in Table 6 for the sake of brevity. The detailed results for all datasets and algorithms used are given in Appendix B.

Dataset	% Accuracy	MAE	RMSE	AUC
1	68.22	0.40	0.45	0.73
2	73.80	0.31	0.44	0.78
3	68.36	0.39	0.44	0.75
4	70.58	0.37	0.43	0.77
5	69.27	0.39	0.44	0.76
6	72.49	0.37	0.43	0.79
7	**73.51**	**0.35**	**0.42**	**0.81**

Table 6 - Summary of logistic regression models trained for assessment evaluation datasets

It is important to note that comparisons between models across the groups of datasets 1-4 and 5-7 must be made carefully if at all. Therefore this analysis will first focus on comparisons within these groups.

Dataset 2 stands out among datasets 1-4 for having a relatively high accuracy rate, however this is confounded by the fact that the model makes use of student-identifying information. In fact, the logistic parameters for this model show that the students who answered incorrectly to all core items have their post-test answers predicted as incorrect. We believe this is an undesired effect of using this information and take this as evidence that the model is overfit to that circumstance.

Dataset 4, which takes advantage of the most information regarding the students' performance in pre-requisite skills of the core items, is the remaining . The paired T-test in Weka demonstrates that the difference between Dataset 1 (skill-identifying and correctness information only) and both Dataset 3 and 4, in the four metrics shown above, are significant. We present this as evidence that considering the students' performance in pre-requisites of the core skills and also remediation work can lead to a better prediction of their core skill knowledge.

In predicting knowledge of each of the skills in the graph, we believe that a similar claim can be made from observing that Dataset 7, the model with the most information regarding completed remediation work, performs with the highest classification accuracy and AUC as well as minimum error and is significantly different from Datasets 5 and 6. The data gathered regarding remediation work is therefore beneficial on some level in predicting student knowledge after a two week period.

We also find it possible to address the question of whether PLACEments is a more accurate assessment of specific skills than a fixed-item benchmark exam. Since Dataset 6 (which does not assume non-core skills are known) also demonstrates a reliably different result from Dataset 5 (which makes that

assumption), we argue that a benchmark test which assumes that previous mathematics skills are known is may be a flawed assessment.

4 Related Work

One commercial system, Knewton, represents mathematics curriculum as a graph of dependent skills and includes an adaptive remediation component, but to our knowledge no work has been published to describe or evaluate that system's per-skill assessment accuracy. Furthermore, the service is focused on college mathematics readiness and remediation rather than K-12 diagnostic benchmarking. [11]

The currently popular Khan Academy video instruction service [14] also employs a conceptually similar graphical structure of skills and dependencies, but we have found no peer-reviewed, empirical evidence of its assessment value either.

5 Conclusions & Future Work

In this project, we have described a novel use of graphical knowledge representation to drive both an adaptive assessment *and* adaptive practice. The PLACEments system was evaluated for its utility and accuracy as a mathematics skill assessment system across a small population of middle school students.

We have shown through comparative analysis of various predictive models that the PLACEments test is overall a reliably better than a fixed-item benchmark test as an assessment of individual skills. Models that include as features the student's performance on test items and/or remediation work for pre-requisite skills result in more accurate predictions of their performance on corresponding post-test items. However this result fails to indicate a causal relationship between knowing a pre-requisite skill and knowing its post-requisite, so we make no such claim.

We have not been able to reason very deeply about the fidelity of the skill dependency graph used in our experiment. We believe that research towards vetting such graphs or methods of learning the skill graph topology could significantly improve the assessment value of the adaptive test.

References

1. Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N.T., Koedinger, K. R., Junker, B., Ritter, S., Knight, A., Aniszczyk, C., Choksey, S., Livak, T., Mercado, E., Turner, T.E., Upalekar. R, Walonoski, J.A., Macasek. M.A., Rasmussen, K.P. (2005). The Assistment Project: Blending Assessment and Assisting. In C.K. Looi, G. McCalla, B. Bredeweg, & J. Breuker (Eds.) *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, pp. 555-562. Amsterdam: IOS Press.

2. Heffernan, N., C. Heffernan, K. Dietz, D. Soffer, J. W. Pellegrino, and S. R. Goldman. "Improving Mathematical Learning Outcomes Through Automatic Reassessment and Relearning." *American Educational Research Association* (2012). Web.

3. Rasch, G. *Probabilistic Models for Some Intelligence and Attainment Tests*. Chicago: University of Chicago, 1980. Print.

4. Embretson, Susan E., and Steven Paul. Reise. *Item Response Theory for Psychologists*. Mahwah, NJ: L. Erlbaum Associates, 2000. Print.

5. Northwest Evaluation Association. *Measures of Academic Progress (MAP)*. Computer software. Northwest Evaluation Association (NWEA), n.d. Web. 3 Dec. 2012. <http://www.nwea.org>.

6. Assessment Technology Incorporated. *Galileo K-12 Online*. Computer software. Assessment Technology Incorporated (ATI), n.d. Web. 3 Dec. 2012. <http://www.ati-online.com/>.

7. Militello, Matthew, and Neil Heffernan. "Which One Is "Just Right"? What Every Educator Should Know About Formative Assessment Systems." *International Journal of Educational Leadership Preparation* 4.3 (2009): n. pag. Connexions. National Council of Professors of Educational Administration (NCPEA), 20 Aug. 2009. Web.

8. Almond, Russell G., Valerie J. Shute, Jody S. Underwood, and Juan-Diego Zapata-Rivera. "Bayesian Networks: A Teacher's View." *International Journal of Approximate Reasoning* 50.3 (2009): 450-60. Web.

9. Belhumeur, Corey. "A Computer Adaptive Assessment That Helps Students Practice in Their Zone of Proximal Development: Automatically Determining Which Skills Students Should Work on at a given Point." Thesis. Worcester Polytechnic Institute, 2012. Print.

10. Embretson, Susan E. "Generating Items during Testing: Psychometric Issues and Models." *Psychometrika* 64.4 (1999): 407-33. Print.

11. Ferreira, J. (2011). How Adaptive Learning Can Keep Every Student on Track to Graduation. In S. Barton et al. (Eds.), *Proceedings of Global Learn 2011* (p. 1440). AACE.

12. Mingyu Feng; Heffernan, N.T.; Heffernan, C.; Mani, M., "Using Mixed-Effects Modeling to Analyze Different Grain-Sized Skill Models in an Intelligent Tutoring System," *Learning Technologies, IEEE Transactions on* , vol.2, no.2, pp.79,92, April-June 2009.

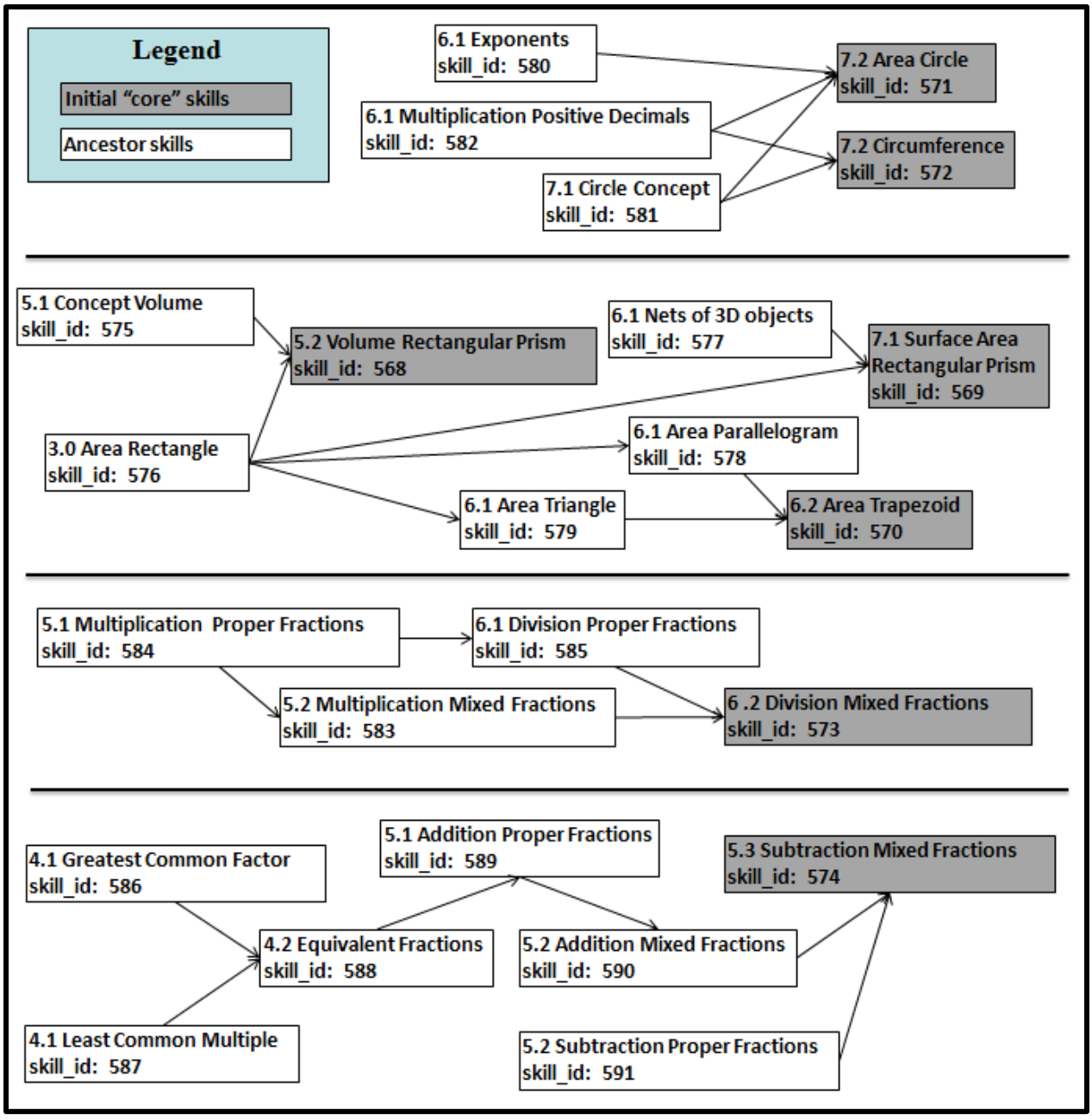
13. Corbett, Albert T., and John R. Anderson. "Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge." *User Modelling and User-Adapted Interaction* 4.4 (1995): 253-78. Print.

14. Thompson, Clive. "How Khan Academy is changing the rules of education."*Wired Magazine* 126 (2011).

15. "Mission Statement." *Common Core State Standards Initiative*. N.p., n.d. Web. 14 Apr. 2013. <http://www.corestandards.org/>.

16. Mitchell, Tom M. *Machine Learning*. New York: McGraw-Hill, 1997. Print.

Appendix A Experimental Skill Dependency Graph



Appendix B Summary of Predictive Models

The table below shows Dataset 7 in comparison with the other six where a paired T-test has been performed in Weka Experimenter to determine significance of the differences between each value for $p<0.05$. Green shading denotes "Dataset 7 performed better than this dataset for the given algorithm in this row," and conversely red shading means the opposite. No shading in a cell implies that there is no significant difference between it and Dataset 7 for that algorithm.

ZeroR is the same as taking the majority class. OneR is a "one-rule" approach where a single attribute is chosen to split the instances and the majority class is taken from each. J48 is Weka's implementation of ID3/C4.5 decision trees, using stock parameters except for minimum leaf size of 5 instances. Logistic is Weka's stock logistic regression.

	Datasets						
Model type	[7]	[1]	[2]	[3]	[4]	[5]	[6]
	Classification Accuracy (%)						
rules.ZeroR	50.46	60.45	60.45	60.45	60.45	50.46	50.46
rules.OneR	69.33	68.54	65.66	68.54	68.54	69.33	69.33
trees.J48	73.48	67.4	60.45	68.97	70.53	69.25	71.91
functions.Logistic	**73.51**	68.22	73.8	68.36	70.58	69.27	72.49
	Mean Absolute Error (MAE)						
rules.ZeroR	0.5	0.48	0.48	0.48	0.48	0.5	0.5
rules.OneR	**0.31**	0.31	0.34	0.31	0.31	0.31	0.31
trees.J48	0.35	0.42	0.48	0.4	0.37	0.39	0.37
functions.Logistic	0.35	0.4	0.31	0.39	0.37	0.39	0.37
	Root Mean Squared Error (RMSE)						
rules.ZeroR	0.5	0.49	0.49	0.49	0.49	0.5	0.5
rules.OneR	0.55	0.56	0.59	0.56	0.56	0.55	0.55
trees.J48	**0.42**	0.46	0.49	0.45	0.44	0.45	0.43
functions.Logistic	**0.42**	0.45	0.44	0.44	0.43	0.44	0.43
	Area Under ROC						
rules.ZeroR	0.5	0.5	0.5	0.5	0.5	0.5	0.5
rules.OneR	0.69	0.65	0.61	0.65	0.65	0.69	0.69
trees.J48	0.8	0.68	0.5	0.72	0.76	0.76	0.78
functions.Logistic	**0.81**	0.73	0.78	0.75	0.77	0.76	0.79

Dataset	Description
[1]	Use 7 core pre-test items to predict 7 core post-test items. Uses only skill info and pre-test correctness.
[2]	Same as [1], but add student-identifying parameter.
[3]	Same as [1], but add total percent of the skill's prerequisites already known based on pre-test data.
[4]	Same as [3], but add total percent of remediation work completed for skill and its pre-requisites.
[5]	Predict each of the 24 skills using skill data and pre-test correctness. Correctness of items for core skills is reported accurately. We manipulate the correctness for non-core skills to all be "correct," to simulate an assumption that may be made when administering a fix-item benchmark (that the students know the pre-requisites).
[6]	Same as [5], except that correctness of non-core skills is reported truly. This correctness is a ternary attribute consisting of "inferred," "incorrect" or "correct." Inferred in this case means the skill was not tested because its post-requisites tested positively.
[7]	Same as [6], but add remediation work data including: number of items completed in remediation, sum of time spent on remediation and ternary attribute telling whether "none was assigned," it was "completed" or is still "incomplete."